6F−1

画像処理用超並列プロセッサ AMP のプログラミング言語Valid-Aの拡張について

山元規靖　鶴田直之　谷口倫一郎　雨宮真人
九州大学総合理工学研究科

1. まえがき

著者らは，画像処理用自律型非同期超並列プロセッサ AMP[1]上で，画像の早期処理から画像理解まで柔軟な記述ができるプログラミング言語 Valid-A[2]を開発している．AMP の各 PE は，外部メモリ(共有メモリ)を持たず，静的データ駆動方式に基づいて構成されている．

Valid-A は関数型をベースとしていたため，状態の保持が必要な処理はループを用いて記述する等のテクニックが必要であった．また，複数のノードで構成される計算体の入力ノードに多重トークンを許すストリームノード[3]を Valid-A で簡潔に表現する必要がある．

今回，Valid-A にモジュールの概念を導入することによりこの問題に対処するとともに，PE 間通信の記述を明確にすべく拡張を行なったので報告する．

2. Valid-A におけるモジュールの導入

前述のように，Valid-A は関数型をベースにしていたため，履歴依存処理においては状態をループによって保持する必要があった．このため，状態保持のループと単なる繰り返しのループが混在し解り辛くなる傾向にあった．この問題を解決し，さらに PE 間通信等を明確に記述するため以下のようなモジュールを導入する．

```
module module_name (x1, x2, ...) return (int,...)
{    states s1=x1, s2=x2, .....
     inports inport_variables
     outports outport_variables
     new (s1, s2, ...) = expression_for_new_states }
```

上記では，引数 *x1, x2, ...*のモジュール *module_name* が定義されている．このモジュールは return で定義される型をその数だけ返す．states は状態保持のための変数とその初期値を定義し，inports，outports はそれぞれモジュール間通信の入力，出力ポートを定義する．入力ポートはストリームノードとして定義される．new 以下は状態の更新を記述するものである．

モジュールも関数配列と同様に配列表現が可能である．このモジュール配列については後で述べるプログラミング例で示す．

以下では上記モジュールについて更に詳しく述べる．

2.1 状態の保持

従来の Valid-A では，状態の保持が必要な処理は以下のようにループを用いて陽に記述する必要があった．

```
function it f(x) return(int) =
  for (s) init (x)
    let y=recieve(LP1),
        j=recieve(LP2),
        z=(s+y)/2
    in if (j == 0) then return(z)
       else recur(z)
```

これは状態を表す変数 s に対して，LP1 で受けとった値 y との平均を新しい s の値とする処理である．このような状態保持のループは単なる繰り返し処理のループと区別がつきにくい．

そこで前述のモジュールを用い上記の例は以下のように記述される．

```
module O(x) return (int)
{    states s=x
     inports y, j
     new s = let z=(s+y)/2
             in if (j == 0) then return(z)
                else z              }
```

このようにモジュールでは，状態の保持をループなどのテクニックを用いずに明確に記述できる．

2.2 モジュール間通信

モジュール間の通信ポートとその接続関係は inports と outports に陽に記述する．以下に通信ポートの記述例を示す．

```
module O1(x) return (int)
{    states s=x
     inports in1 from O2.out1, in2 from O2.out2
     outports out1 to O2.in1, out2 to O2.in2
     new s = expression1                 }
```

Programing Language Expanded Valid-A for AMP

Noriyasu YAMAMOTO, Naoyuki TSURUTA, Rin-ichiro TANIGUCHI, Makoto AMAMIYA
Department of Information Systems, Graduate School of Engineering Sciences, Kyushu University

```
module O2(x) return (int)
{    states s=x
     inports in1 from O1.out1, in2 from O1.out2
     outports out1 to O1.in1, out2 to O2.in2
     new s = expression2 }
```

二つのモジュール O1 と O2 の out1 と out2 がそれぞれ互いの in1 と in2 に接続されている．to と from 以下は接続されているモジュールとそのポートを示し，複数のポートを記述できる．また，ポート間の接続は対応する inports，outports のどちらか一方に明記されていれば良い．

このように，モジュール間通信は，inports，outports においてポートの接続関係が記述されるため分かりやすいものとなっている．

2.3 モジュールへのデータの割り付け

モジュールは下記の map 式によりデータが割り付けられ起動される．

map *module_name* initial_states (*x1, x2,...*)

module_name がモジュール配列の場合，map 式により割り付けられるデータ *x1, x2, ...* もそれに対応した配列型でなければならない．map 式は，データ *x1, x2, ...* によりモジュールを起動し，モジュールの戻り値を返す．

3. プログラミング例

ここでは前述のモジュールを用いたプログラミング例として 2 値画像の細線化を挙げる．

ここで挙げる細線化は，画素の 8 近傍に注目して，消去可能な画素は消去するという処理を繰り返すもので[2]，その間，各々の 8 近傍の画素の値を保持しなければならない．そこで，8 近傍の画素値を示す変数を states で定義し，下記のようにプログラムを記述する．

— **Program** [細線化] ————

```
function main() return(signal)
  = let x=load_image(),
        z=map THIN initial_states(x),
    in  display_image(z) ;

array_of_module THIN[256][256](pix)
 return(int) for_each_module[i][j]
{ if (i>0 & i<255 & j>0 & j<255) then
 {states   xa1=x1, ..., xa8=x8
  inports  x1 from THIN[i][j-1].pix,
           ........................,
           x8 from THIN[i-1][j-1].pix,
           xb1, ..., xb8, xc1, ..., xc8
  outports pix,
           y1 to (THIN[i][j-1].xb5,
                  ..., THIN[i-1][j-1].xb4),
           y2 to (THIN[i][j-1].xc5,
                  ..., THIN[i-1][j-1].xc4)
  new(xa1, ...,xa8)
  = if (pix==1) then
      let Mask=array(256, 0, 1, 0, 2, ...),
          y1=Mask[Comp(xa1, ...,xa8)],
          xy1=if (xa1==1) then xb1 else xa1,
          ..............................
          xy8=if (xa8==1) then xb8 else xa8
      in if (y1==1) then
          let y2=Mask[Comp(xy1, ...,xy8)],
             xz1=if (xy1==1) then xc1 else xy1,
             ..............................
             xz8=if (xy8==1) then xc8 else xy8
           in if (y2==1) then (xz1, ...,xz8)
              else if (y2==0) then return(0)
                   else return(1)
          else if (y1==0) then return(0)
               else return(1)
        else return(1) ;
  ..................................... }
function Comp(x1,...,x8) return(integer)
 = let xx1 = if (x1==0) then 0 else 1,
       ..........................
       xx8 = if (x8==0) then 0 else 1
   in  xx1<<7|xx2<<6|xx3<<5
         |xx4<<4|xx5<<3|xx6<<2|xx7<<1|xx8 ;
```

上記の outport のように通信ポートの接続関係は，一つのポートから複数のポートへの記述ができる．最初の main 関数は，二値画像を取り込み(load_image)，取り込んだデータを map 式によりモジュール THIN に割り付けて起動し，その細線化された画像を表示する(display_image)ものである．

関数配列のみを用いた場合と比較して(参考文献[2]参照)，PE 間通信の接続関係が明確になっており，また，ループを用いたテクニックを使用せずにすっきりと状態を保持する処理が記述できている．

4. むすび

本稿では，AMP 上のプログラミング言語 Valid-A にモジュールの概念を導入することにより，状態の保持が必要な処理を特別なテクニックを用いずに記述でき，また，PE 間通信の記述法もより明確になることを示した．

Valid-A では，状態の保持が必要な処理や PE 間通信を含む処理はモジュール(配列)を使用し，それ以外は通常の関数(配列)を使用するのが望ましい．これにより，簡潔で分かりやすいプログラムが記述できる．

参考文献

[1] 谷口，雨宮："画像処理と理解のための自律型非同期超並列プロセッサ AMP"，「画像理解の高度化と高速化」シンポジウム講演論文集，pp.53–58(1989).

[2] 山元，鶴田，谷口，雨宮："画像処理用超並列プロセッサ AMP のプログラミングと性能評価について"，情報処理学会論文誌，Vol.32，No.7，pp.933–940(1991).

[3] 趙，川野，谷口，雨宮："画像処理用超並列プロセッサ AMP における Simplified Stream Buffer の実現"，情報処理学会第 44 回全国大会 (1992).